Datacomputer and SIP Operations: ~~1979~~ Final Technical Report

LEVEL

(12)

R. Mark Chilenskas
Matthew Maltzman
Anil K. Nori
Joanne Z. Sattley
Nancy I. Wolfe

Technical Report
CCA-80-01
February 20, 1980

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139

(12)

Datacomputer and SIP Operations Quarterly Technical Report

Technical Report
(14) CCA-80-01

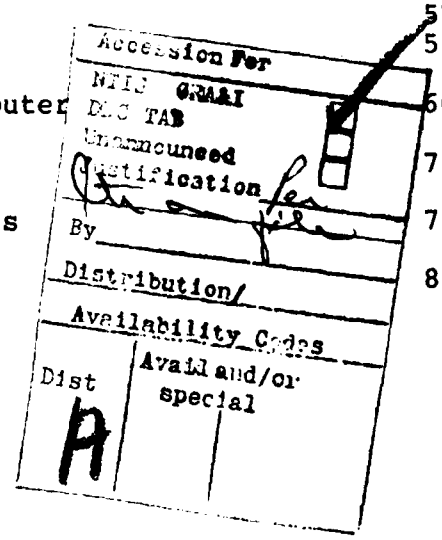
R. Mark Chilenskas
Matthew/Maltzman
Anil K./Nori
Joanne Z./Sattley
Nancy I./Wolfe

February 20, 1980

(15) N00039-79-C-0203

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. N00039-79-C-0203, ARPA Order No. 3540. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Table of Contents



1. Summary

Computer Corporation of America continued to provide Datacomputer and SIP service to the ARPAnet community throughout the final quarter of 1979. The Datacomputer is a network-oriented database management utility which is used in diverse general networking applications. The Datacomputer's software system is designed to allow convenient and timely access to large on-line databases and to promote data sharing by multiple remote users communicating over a network. → page 4

The Datacomputer offers cost-effective large-volume storage capacity plus the ability to select and retrieve manageable subsets of data for efficient local processing. It is the only operational general purpose database system capable of manipulating data sets in excess of a trillion bits.

This very large storage capacity is made possible by the incorporation of an Ampex Terabit Memory System (TBM). The TBM at CCA is the only public installation of a data storage system based on videotape technology. The site is configured to hold up to 175 billion bits on line,

distributed over four TBM tape drives. Storage of data which may be referenced very seldom -- perhaps once in several years -- is managed on off-line tapes.

CCA operates and maintains the Datacomputer; as an attendant obligation, it responds quickly to the needs and problems of its users when they arise. The support functions include the operation of an on-line netmail help facility and 24-hour coverage of a telephone trouble line.

During this reporting period in particular, assistance was given to a set of potential Datacomputer users who were accustomed to designing CODASYL-type databases -- and who now were especially interested in developing a quasi-relational database using the Datacomputer. As a by-product of this effort, a mapping rule for translating network datamodels into relational datamodels was devised. A description of the algorithm is given in Appendix A.

Underlying the design of the Datacomputer software is the expectation that the direct user of the Datacomputer will be a program running on a remote network host. CCA has implemented and continues to maintain a number of programs and subroutine packages which run on several ARPAnet hosts; these programs provide convenient access to the Datacomputer for remote users across the ARPAnet.

The interfaces provided by these programs range in scope from very low-level utility functions (which enable user programs to treat the Datacomputer as if it were a simple input/output device) to very sophisticated applications which exercise the Datacomputer's extensive indexing capabilities. Examples include the following: the RDC program for on-line input of Datacomputer instructions; the DCPKG subroutine package for interfacing FORTRAN and COBOL -- and now PASCAL -- programs to the Datacomputer; and the MARS Message Archiving and Retrieval Service. MARS users do not even communicate with the Datacomputer-interface programs. Rather, they use ordinary network mail to transmit messages for filing and retrieving; "demon" programs operating as background processes on CCA-Tenex handle the actual storing and retrieving of messages on the Datacomputer.

In 1979, the DOCFile Document Filing and Retrieval Service was developed along the lines laid out by MARS, but distinct in that the document system performs direct file transfers over the network rather than relying on the mail delivery system. DOCFile has been installed on the USC-ISIE and CCA-TENEX computers.

Recent discussions at the MIT Laboratory for Computer Science regarding the use of a document control system for

an office-modelling project led to the preparation of a report on a low-level interface to DOCFile. The specifications for this interface are included in this report as Appendix B.

The seismic application was heretofore the largest user of the Datacomputer. This was true not only in terms of the amount of data stored, but also in terms of its complexity and the required bandwidth of transfer. CCA developed and operated a dedicated Seismic Input Processor (SIP) to field the real-time input stream. Both the real-time data stream storage via the SIP and the SRO files storage from the Albuquerque Seismological Laboratory were shut down during this reporting period.

Maintenance work on the Datacomputer itself, funded in part under Contract N00039-78-C-0443, engendered two new program releases, Versions 5/4 and 5/5. The usual rigorous pre-release testing procedures were followed to ensure the upward compatibility of the modifications.

2. Support of the User Community

An important task for CCA under this contract has been to respond to queries about the Datacomputer from both the seismic and general user communities. Because the Datacomputer is a unique -- and evolving -- network resource, consultations, coordination, and technical assistance are necessary if the best use is to be made of it. In the following subsections we give overviews of the seismic and general user communities and describe the significant developments in three of the CCA-provided Datacomputer user interface programs.

2.1 The Seismic User Community

The seismic users of the Datacomputer are primarily (but not exclusively) involved with research in the area of monitoring underground nuclear tests. The data filed prior to 31 December 1979 continues to remain available for retrieval, although the filing activity has ceased.

The main source of seismic data is the worldwide network of seismic instruments, transmission lines, and data processors known as the VELA Network. Some of the components of the VELAnet are on the ARPAnet and use it as a data transmission system. Other parts of the VELAnet use leased lines for real-time data or rely on physical shipment of magnetic tapes for non-real-time data.

The Datacomputer is the primary storage and retrieval resource in the VELAnet. This activity requires storage of very large amounts of on-line data including the following:

- readings from arrays of seismic instruments, with real-time processing of some of the readings;
- seismic readings from individual Seismic Research Observatories (SROs) which are forwarded from the Albuquerque Seismological Laboratory;
- status and calibration information on instruments and on the VELAnet;
- derived seismic event summary information; and
- extracted signal waveforms corresponding to events.

The seismic instrument readings can be further divided into long period (one sample per second) and short period (ten or twenty samples per second) while the event summary information can be divided into preliminary and final versions. Since the Datacomputer is not designed to receive real-time data, a special dedicated miniprocessor, the SIP, was used to buffer it as described in Section 4 below.

The status and event summary data are relatively compact and are of sufficient importance that they are kept on line. The seismic data readings, however, are so voluminous that they fill many reels of mass storage system video tape -- 14 at last count. Though most of these reels are, of necessity, kept off line, they can be mounted when needed on a day's notice.

In the last half of 1979, CCA provided consultation and assistance to the seismic user community in utilizing the Datacomputer in the following noteworthy ways:

. From June through August, CCA assisted the Seismic Data Analysis Center (SDAC) in several ways in connection with a series of demonstrations they were planning for the State Department. CCA prepared and sent various demonstration materials -- a TBM tape, slides showing the CCA site, copies of several technical reports

-- provided stable system operations during the pre-demo tests as well as during the demo, and consulted on the formulation of efficient Datalanguage for performing various retrievals.

. In September and October, CCA investigated the techniques used for acquiring the SRO station tapes and alternate ways of storing the data on the Datacomputer. A report in the form of a net message, including funding requirements for the alternatives, was mailed to ARPA/NMRO.

. In November, CCA responded to a request from visiting Norwegian seismologists for information on how to access the seismic data on file in the Datacomputer.

. Following the termination of tranmissions from the CCP, the SIP was powered down. Although no new data will be stored, the existing files continue to remain available for retrieval by the seismic community for a limited period of time.

During this reporting period, it was decided that the Ampex mass storage system would be phased out by 31 March 1979. After that date the seismic files will no longer be available.

2.2 The General User Community

There are three broad categories of non-seis
Datacomputer usage. These are:

- the on-line interactive interface programs;
- the program demons, and
- the special-purpose asynchronous-access application packages.

At present, on-line interactive use of the Datacomput
the first category above, consists of the DFTP and
programs. DFTP connects in real time to the Datacompu
and enables its users to transmit local files into
from the Datacomputer. The RDC program is also
real-time interface program; it is used mainly a
debugging tool for testing and refining Datalangu
instructions for later automated use by higher-le
programs. RDC is used heavily by the CCA staff
pre-release reliability and regression testing of
Datacomputer versions.

There are a number of programs that automatically st
data into the Datacomputer. Among these are the SUI

system at MIT-DMS which stores information derived by polling the status of the ARPAnet server hosts, and the IMP-LOGGER system at BBN which stores information on events in the ARPAnet communications backbone.

The more sophisticated applications, MARS and DOCFile, provide local interactive programs so that users may queue requests for archiving and retrieval services. The mail archiving activity, in particular, has stimulated mail system developers to include the concept of an archive button in their programs -- so that, by means of a single keystroke, users can designate that a copy of the message be sent to the archives. HERMES users have but to set up their templates appropriately in order to achieve a similar effect.

Although in the past it was possible to keep all non-seismic data in the Datacomputer on line, general usage has now grown to such an extent that an archival general purpose mass storage tape has had to be allocated and some rarely referenced files have been transferred to it. This archival tape is not always mounted.

Following the termination of TBM-based Datacomputer service at the end of March 1979, a disk-based Datacomputer will remain available for retrieval of public information which is of interest to the ARPAnet community

-- such as the Internet Experiment Notebook, RFCs, MsgGroup transcripts, and NSW documents.

2.3 User Interfaces

The Datacomputer is designed to be accessed only via ARPAnet connections; hence, communication with the Datacomputer requires a process running on some network host to handle the other end of the connection. CCA supplies and maintains a collection of subroutine packages and programs which provide easy access to the Datacomputer. Three of these programs are discussed in this section; included here are reports on extensions to the DCPKG program, on MARS Service and on the DOCFile System.

2.3.1 The DCPKG Subroutine Package

DCPKG provides Datacomputer-communication handling utility routines for inclusion in users' programs. Previously, software support was available for accessing the Datacomputer from FORTRAN and COBOL programs. As a mark of progress, we can now provide access to the Datacomputer

from Pascal programs. This section discusses this facility.

Approach:

Conceptually, there were two approaches that could have been taken to implement the Pascal interface:

1. write a shell-type of interface package utilizing existing CCA software;

2. extend the existing Fortran/Cobol interface package DCPKG to include the Pascal interface also.

We have taken the second approach for its ease of implementation. The extensions (requiring additional code) to DCPKG would ordinarily be necessary for recognizing the Pascal calling conventions. However, Pascal implementations provide a feature which completely eliminates the need for these changes to DCPKG. This feature allows the simulation of Fortran calling conventions from Pascal programs, a very easy task in Pascal.

Thus by simulating Fortran calling conventions, Pascal programs can access the Datacomputer as if the Datacomputer were being accessed from a Fortran application program. A more detailed discussion and a sample Pascal program are given in Appendix C.

2.3.2 MARS, a Message Archiving and Retrieval Service

MARS is a package of programs which, working cooperatively, provide a unique way for ARPAnet correspondents to store messages on the Datacomputer, and to later retrieve selected subsets of the archived correspondence. Using the inverted-list indexing capabilities of the Datacomputer, each message is indexed on the field-names and keywords found in its header (separately by recognized fields), by date, and by words in the text-body; it may be retrieved by Boolean combinations of any of these.

The interactions between MARS users and the Datacomputer are accomplished by means of standard ARPAnet messages; the user h(im/er)self does not participate directly in the transmissions between MARS and the Datacomputer (except, of course, for providing the input and the motivation).

Standard mail-handling programs such as MSG and SNDMSG have been used for archiving individual pieces of mail. For archiving several messages -- or arbitrarily large collections of mail -- as a single operation, CCA

developed the MBQ program. MBQ constructs batch-mail input files and queues them for delivery to the MARS-Filer program operating on CCA-Tenex. A further description of its operating characteristics is given in Appendix D.

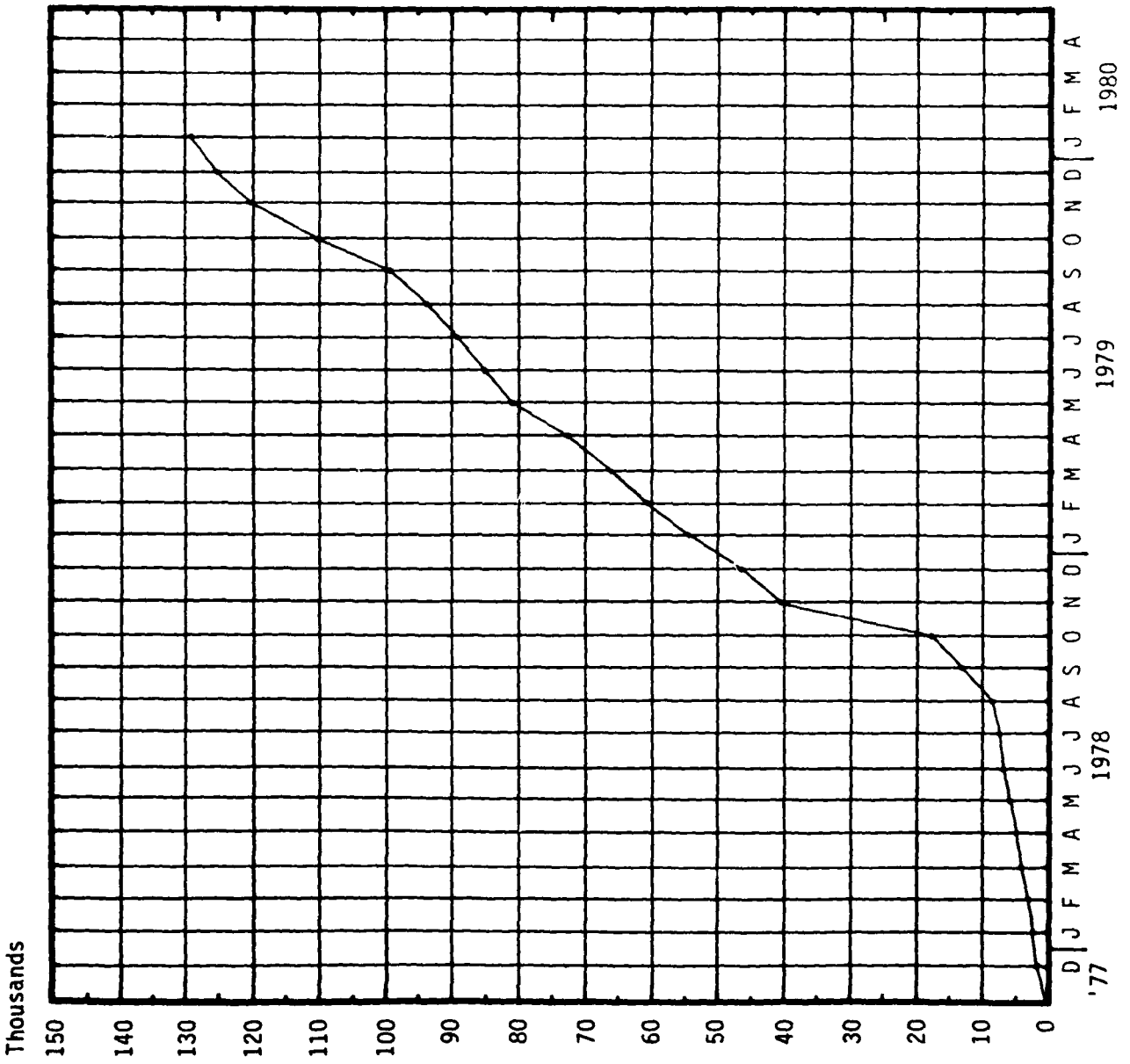
Retrievals also are triggered by messages, and it is possible to use any available message-composing program for this purpose. Appendix E contains the essential MARS instructions for performing retrievals.

In addition, many network hosts offer the interactive RR program which is designed to assist users in preparing query-messages (also called "RRs", standing for Retrieval Requests). These messages, when delivered to the MARS-Retriever program operating on CCA-Tenex, are translated into Datalanguage requests which are then transmitted to the Datacomputer. The retrieved messages are mailed back to the requester and will appear as new mail in h(is/er) message file.

The basic plan of MARS operations was originally distributed as Network Working Group RFC #744, NIC #42827. The first release of a functional package was described in 1978 as an informal report, MARS-Note #1.

The growth of the message database is displayed below in Figure 2.1, a Cumulative Summary of Messages Filed through January 1980.

Mars: Cumulative Summary of Messages Filed
Figure 2.1



2.3.3 DOCFile, a Document Storage and Retrieval System

The DOCFile system is the latest Datacomputer application to be developed by CCA. It was designed to repond to the needs of ARPA and contractor personnel for a large-volume document management facility. The requirements were for a system which could be used to store selected types of official documents and subsequently to retrieve them on the basis of a flexible set of retrieval criteria -- for example, by author name or by contract number.

The initial specification called for the following document types:

- Technical Reports
- Final Reports
- Software Sources
- Proposals (without pricing information)
- MRAOs

And we are considering additional types, such as:

- IENs, and
- RFCs

The DOCFile system, as seen by its user, is a friendly menu-driven program which interacts with the user to construct requests for a set of actions to be performed. (Retrieving a document is an example of an action.) Each action may be assigned priority (high, medium, low) by the user. The requested actions are appended to a work-list which may be examined by the user at will.

The DOCFile system, in operation, is made up of an interactive user program and a background demon job. The demon program periodically scans the work-list, performs the requested actions in priority order, and posts the results back on the work-list. Acknowledgement of some actions will, in addition, be reported by net-message in the user's mail file.

The queue-level interface specifications for the DOCFile system are given in Appendix B.

3. Datacomputer Development

The Datacomputer is an operational system in which some evolutionary development is occurring. Version 5/5 of the Datacomputer was put into service in December 1979, and is discussed further below.

For additional details on the structure of the Datacomputer and its development history, the reader is referred to Technical Report CCA-79-11, Datacomputer and SIP Operations: 1978 Final Technical Report. We continue to receive and respond to requests for copies of this report and the Datacomputer User Manual.

Version 5/5 of the Datacomputer was installed on 19 December 1979. It incorporates a number of maintenance changes and an update of the subjob initialization logic. The maintenance changes, funded in part by the SDD-1 ACCAT contract N00039-78-C-0443, include:

- Fixing the automatic backup routine to properly reinitialize its flags.
- Removing the "FILE OPEN A LONG TIME!" error condition.

- Exempting the UTILITIES subfork from timeout and suspension.

- Ignoring CLOSF errors when a separate data connection open fails.

The initialization routines were modified to generate subjobs dynamically as needed, rather than having all possible jobs started at once. This was done primarily for the SDD-1 environment where there is a potential for many subjobs at peak load but the normal load is substantially smaller. In addition, crashed jobs are restarted, which prevents a Datacomputer from being shut down by a bug in the SDD-1 user programs.

The subjob changes included adding a jOBS command setting the minimum and maximum number of jobs to remain available and the maximum number of jobs ever allowed. This permits site tailoring without patching the Datacomputer code. It is useful in a user environment when the resources allocated to the Datacomputer must be temporarily limited without changing its priority (e.g., if the total resources to the machine are temporarily limited or the load is higher than normal).

Within the last few years, the Datacomputer program gradually grew in size until it reached the point where it

completely filled the machine's 256K-word address space. At this point, adding more features to the Datacomputer required that it be broken up into two separate processes, or forks, so that the resources of two complete memory spaces would be available. The decision was made to put all of the Datacomputer's utility routines in one fork and all the rest of the Datacomputer in the other. The main advantage of this approach is that the utility routines are used relatively infrequently, and as a result there is little overhead cost in switching from one fork to another.

During the next reporting period CCA, at ARPA's request, will be phasing out the TBM-based Datacomputer and installing a disk-based system in its stead.

4. The SIP

The SIP is a dedicated minicomputer communications system developed and operated by Computer Corporation of America. Until 31 December 1979 it served as an interface between real-time seismic information from the VELAnet and the Datacomputer.

Below we give a general description of the SIP and a summary of its operation during 1979. The general description given is similar to that in our Final Technical Report for 1978. Those familiar with the operation of the SIP in the VELAnet may safely skip it.

4.1 General Description

A primary function of the world-wide VELA Seismological Network (VELAnet) is the collection of real-time seismic data from arrays of seismometers. This data is sent over leased lines and the ARPAnet to the Communications and Control Processor (CCP) at the Seismic Data Analysis Center (SDAC) in Alexandria, Virginia. From the CCP this

data is immediately distributed to various processors and to the Datacomputer, via the SIP, for storage.

The components of the VELAnet that handle real-time seismic array data, except for the Datacomputer, are dedicated systems designed to receive data in real time. The Datacomputer, however, operates within a non-real-time operating system and serves the general ARPAnet community as well as the VELAnet. Furthermore, it is occasionally unavailable due to scheduled and unscheduled maintenance work. To isolate the Datacomputer from these real-time requirements, the SIP was implemented to receive real-time data from the CCP, to buffer and reformat this data on disk, and to periodically forward the data to the Datacomputer.

The SIP is implemented on a DEC PDP-11/40 computer. It has an ARPAnet interface, two RP04 disk drives for buffering data, an operator's terminal, and a status display screen. With the present bandwidth of data being sent over the network to the SIP and the present structuring of the SIP's disk storage, about two days of data can be held by the SIP.

Besides processing seismic data, the SIP software provides for operator communications between itself and the CCP. It also sends messages to the CCP for each chunk of data

when the data has been properly filed in the Datacompu

4.2 SIP Operation in 1979

The SIP operated successfully throughout 1979; it powered down following transmission of the final segm of December 1979 data.

During the course of the year, changes were made in design of the SIP directory structure to raise the max number of hours of data that could be stored on a pack from 32 to 128. The increase was due partly to directory design change and partly to the the cessatio the short period data and the data from the LASA site.

Another change to the SIP, operationally motiva included an improvement in the manner in which ini synchronization was achieved between the SIP inte clock and the CCP time source; the net effect wa speedier synchronization.

5. The Datacomputer Environment

The Datacomputer runs as a user job on the CCA-TENEX ARPAnet host computer under the TENEX operating system. It is an unusually large and complex job composed of "subjobs" most of which serve remote users. Many modifications have had to be made in TENEX to accommodate the special requirements of the Datacomputer and the special hardware in use on the CCA Datacomputer system. The most prominent of these pieces of hardware is the Ampex TBM Mass Storage System. During this reporting period, it was determined that the TBM hardware would be phased out by 31 March 1980.

Below we give a general description of the TBM system, a general description of the modifications we have had to make in the TENEX operating system, and a discussion of TBM and CCA-TENEX hardware.

5.1 The TBM Mass Storage System

The CCA Datacomputer is equipped with the first public installation of the Ampex Terabit Memory (TBM) System. This device uses videotape technology to achieve a maximum on-line capacity of three trillion bits on up to 64 tapes. Maximum data transfer rate is 5.3 million bits per second.

The TBM at CCA is equipped with two dual tape transport modules so at most four tapes, or 175 billion bits (22 billion bytes) can be available on line. All equipment except for the four tape transports is non-redundant in the CCA configuration. This includes one Transport Driver (necessary for a tape to be in motion), one Data Channel (necessary to encode or decode digital information to and from the broadband analog signal on tape), one System Control Processor to coordinate and direct the other units, and one Channel Interface Unit that connects the TBM system to the Datacomputer's PDP-10 system. All of these units, which are non-redundant in the present CCA configuration, must operate properly for the TBM system to be usable by the Datacomputer.

5.2 TENEX Modifications

The modifications and additions that had to be made to CCA-TENEX to accommodate the Datacomputer include changes to both the operating system itself and to a number of separate utility programs that are not part of either the operating system or the Datacomputer proper. These modifications and additions include the following:

. improved efficiency in the network interface code for high volume file-transfer-like data streams sent and received by the Datacomputer;

. device code for using the Ampex TBM Mass Storage system and a set of CalComp 3330-equivalent disks which are used for "staging" -- intermediate storage between the PDP10 memory and the TBM;

. additional statistics-gathering code to aid optimization and analysis of operating system performance;

. a separate network server program augmented to provide status output on the Datacomputer;

. a utility program to run under TENEX for assisting in TBM maintenance;

. a utility program which runs all the time as a background job in CCA-TENEX, monitoring various system resources and alerting CCA personnel in case of problems; and

. additional utility programs for various purposes ancillary to the Datacomputer.

In January 1979, the CCA-TENEX system was modified to make full use of an additional RP-02 disk drive that had been purchased the previous year.

A. Schema Translation: A Case Study

A.1 Introduction

This appendix describes a mapping rule which was devised by Anil Nori of CCA for translating network datamodels into relational datamodels.

Several such mapping algorithms have been proposed in recent years [BORKIN, KAY, MCGEE]. Although this work presents few new ideas in schema translation, the main objective here is to demonstrate the feasibility of a network - relational model mapping in the context of a specific real world situation. Below we present the mapping of a network schema, modelled on DBMS-20 [DEC], to a relational schema on the Datacomputer [CCA].

Part A.2 briefly reviews the network and relational datamodels. Part A.3 describes the mapping algorithm, and applies it to the network schema of the presidential database [TAYLOR AND FRANK]. Part A.4 presents the relational schema of the Presidential database as implemented on the Datacomputer.

A.2 Datamodels

In this section we briefly describe the network and relational datamodels through examples. However, we do not consider the art of database design here. A detailed introduction to network and relational models can be found in [DATE]. As a first step towards database modelling, the user performs the requirements analysis of the real world problem and decides on which objects to represent in the database. An object is made up of one or more data items, called components, and is uniquely identified by a subset of these components called key components. Each object in the database corresponds to an object in the real world being modelled. After the real world objects are modelled, relationships among these objects are modelled. Generally, relationships among objects exist as one-to-many (1:n) and many-to-many (m:n) relationships.

A.2.1 Network Datamodels

Real world objects are represented by record types and their relationships by set types. Particular instances (occurrences) of a record type and a set type are called records and sets respectively. Records represent instances of real world objects, and sets represent relationships among these instances.

Generally, network datamodels are presented using Data Structure diagrams [BACHMAN], which we call network schema graphs. These graphs use a rectangle and an arrow as fundamental components. A rectangle enclosing a name denotes a record type, and an arrow between two record types represents a set type between the two record types. The record type at the tail of the arrow is called the owner record type, and the record type at the head of the arrow is known as the member record type. These arrows are named and they represent the corresponding set type names. The components of a record are usually suppressed in its description.

A set type, also known as owner-coupled set type, models a one-to-many (1:n) relationship between the owner record

type and the member record type, respectively. That is, an owner record can own zero or more occurrences of another record type. Figure 1 presents the Presidential database using network schema graphs.

In figure 1, set type ELECTIONS-WON is an owner-coupled set type between owner record type PRESIDENT and member record type ELECTION. This models a 1:n relationship between PRESIDENT and ELECTION. A president can win more than one election, but a single election can not be won by more than one president. However, consider the relationship between PRESIDENT and CONGRESS. A president can serve more than one congress and a congress can be served by more than one president. This is an example of a many-to-many (m:n) relationship. Since owner-coupled sets typically model 1:n relationships, m:n relationships are generally modelled by introducing a new record type (in this case CONGRESS-PRES-LINK) whether or not this new record type is meaningful to the user. However, in some cases it can contain information which is not relevant to the individual record types but relevant to the relationship between the record types. In figure 1, the m:n relationship between PRESIDENT and CONGRESS is achieved through two 1:n relationships PRESIDENT-SERVED and CONGRESS-SERVED. Record type CONGRESS-PRES-LINK plays member role in both these set types.

Set types whose owner record is SYSTEM are called Singular Sets. Singular sets have only one set occurrence since their owners (SYSTEM) have single occurrence. Generally, these sets provide entry points into the database, in the sense that no other record need be accessed before the first member record of the set is accessed. Singular sets are also used to collect all records of a particular type for sequential access.

In the next section we present a formal model of the network schema graphs.

A.2.J.1 Formal Notation

Formally, a network schema graph N is represented as a collection of record types and owner-coupled set types. Each record type is described as $O(A_1,A_2,...A_n)$ where O is the name of the record type and $A_1,A_2,...,A_n$ are the components of that record type. C(O) is the set of components of the record type O, and Key(O) denotes the key components of the record type O. An owner-coupled set type S_i between an owner record type O_i and a member record type O_{i+1} is represented as a triple $<S_i,O_i,O_{i+1}>$.

A.2.2 Relational models

In a relational model, each real world object is modelled as a relation. Given sets $D_1, D_2, \ldots, D_n$, a relation may be defined as a set of n-tuples each of which has its first component from domain D_1, its second component from domain D_2, etc. A tuple in a relation is uniquely identified by its key components. Often one or more components of a relation will correspond to key components of another relation. Such a component is called a foreign key. Relationships between two real world objects (relations in this model) are modelled with the help of foreign keys.

For example, figure 2 presents many-to-one and many-to-many relationships in a relational model. In the figure, COURSE and CLASS relations participate in a 1:n relationship. A COURSE can be scheduled as more than one class, but a CLASS can have only a single course number. This 1:n relationship is modelled through the foreign key C# (course#) in the CLASS relation. Relation ENROLLMENT models a m:n relationship between CLASS and PUPIL relations. A CLASS can have more than one PUPIL, and a PUPIL can enroll in more than one CLASS. This m:n

relationship is achieved through a third relation ENROLLMENT. Note that, in this example, ENROLLMENT has information component Grade which is not relevant either to CLASS or to PUPIL, but to their relationship.

A.2.2.1 Formal Notation

Formally, a relational model is represented by a schema S which is a collection of relations. A relation is described as $R(A_1,A_2,...,A_n)$ where R is the name of the relation and $A_1,A_2,...A_n$ are the components of the relation R. C(R) denotes the components of the relation R. Key(R) denotes the key components of the relation R which uniquely identify a tuple of R. A relation R_i refers to another relation R_j, iff $key(R_j) \subset C(R_i)$.

A.3 Mapping

In this section we present a mapping from network datamodels to relational datamodels. Then using this mapping we translate the network schema of the presidential database to a corresponding relational schema. Finally, in the next section, we present the

relational schema of the presidential database on the Datacomputer.

A.3.1 Algorithm

For every O_i in the network schema N, there is a corresponding relation R_i in the relational schema S such that

$C(R_i) = C(O_i) \cup \{ key(O_j) \mid \text{for some } s, \langle s, O_j, O_i \rangle \in N \}$
and $Key(R_i)$ is $Key(O_i)$,

for every manual set type [3, 5, 8] S_m, represented as $\langle S_m, O_m, O_{m+1} \rangle$, there is a relation R_m such that

$R_m = Key(O_m) \cup Key(O_{m+1})$
and $Key(R_m)$ is $C(R_m)$.

This mapping algorithm is in no way exhaustive; it does not include all of the features (e.g. MANDATORY and OPTIONAL, SORTed record instances in a set occurrence, etc.) of the network schema being translated. The algorithm essentially translates a data structure diagram of a network schema into a relational schema. These additional features can be enforced externally once the relational schema is available.

A.3.2 Application of the mapping algorithm

Since singular set types are used only as entry points to the system or for sequential access of the member records, there is no corresponding relationship in the real world (there is no object in the real world corresponding to record type SYSTEM). In the same way, singular sets are not modelled in the relational model explicitly. The need for a singular set is alleviated in the relational model by the mere existence of the relation corresponding to the member record type. The modified network schema for the presidential database, without singular set types, is illustrated in figure 3. This schema graph can be formally represented as follows:

N = {

(ADMINISTRATION, CONGRESS, CONGRESS-PRES-LINI
ELECTION, PRESIDENT, STATE);

(<ADMINISTRATION-HEADED,PRESIDENT,ADMINISTR/
<ADMITTED-DURING*,ADMINISTRATION,STATE>,
<CONGRESS-SERVED,PRESIDENT,CONGRESS-PRES-L
<ELECTIONS-WON,PRESIDENT,ELECTION>,
<NATIVE-SONS,STATE,PRESIDENT>,
<PRESIDENT-SERVED,CONGRESS,CONGRESS-PRES-L

}

Now let us apply the mapping to the above network sch
For every record type in N, there is a correspon
relation in S. For example, consider PRESIDENT rela
(corresponding to PRESIDENT record type):

PRESIDENT = Components of PRESIDENT record type
U Key(STATE) since
<NATIVE-SONS,STATE,PRESIDENT> E N.

Since the set type ADMITTED-DURING is a manual set t
we have a corresponding relation ADMINISTRATIONS-ST
defined as:

= Key(ADMINISTRATION) U Key(STATE).

---

*Admitted-During is a manual set type.

After the application of the mapping to the network schema for presidential database, the corresponding relational hierarchy is presented in figure 4.

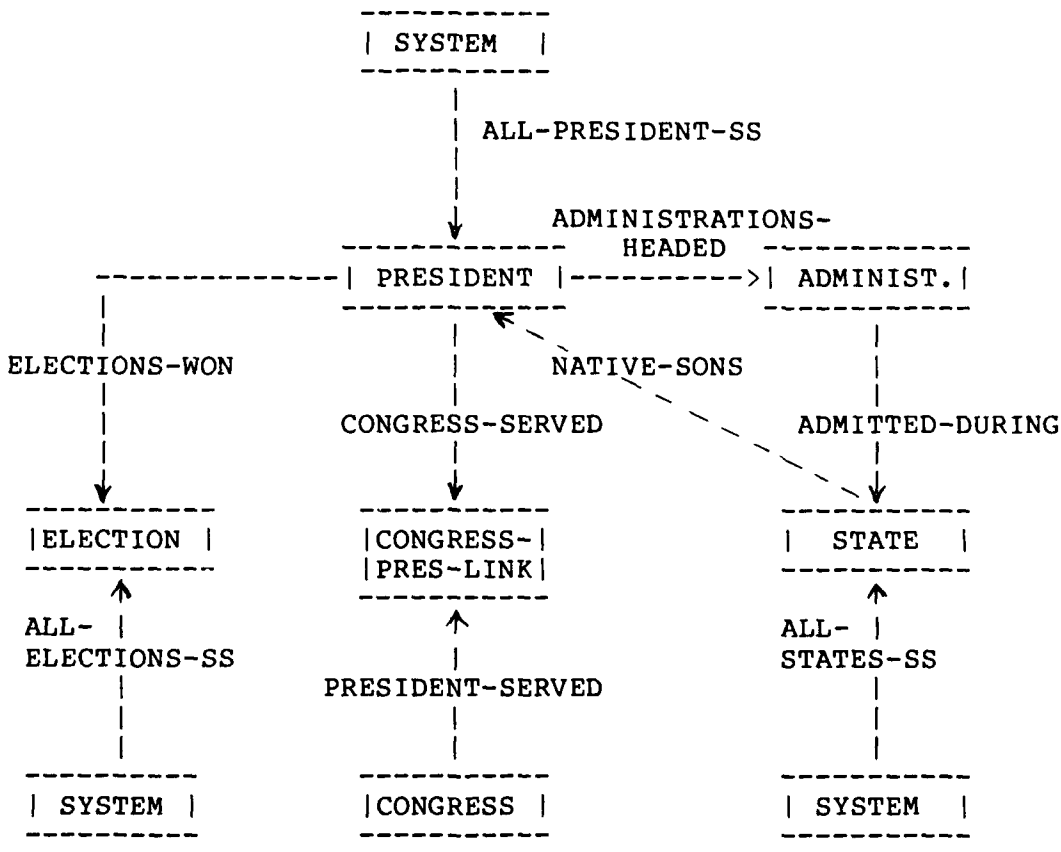
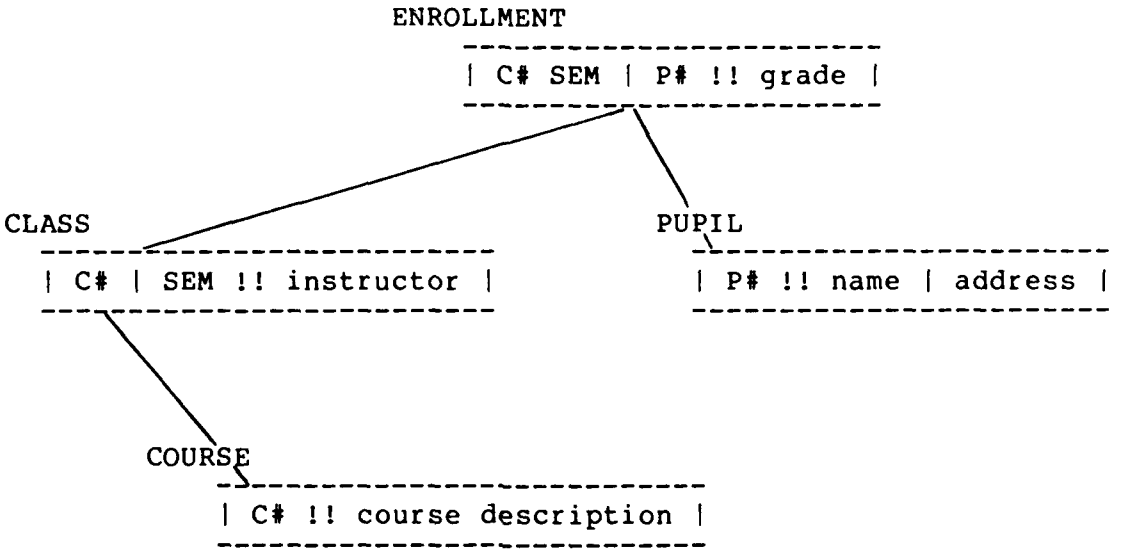


Figure 1. Data structure diagram of presidential database.



note: keys are delimited by !!

Figure 2. Relational schema of Class - Enrollments database.

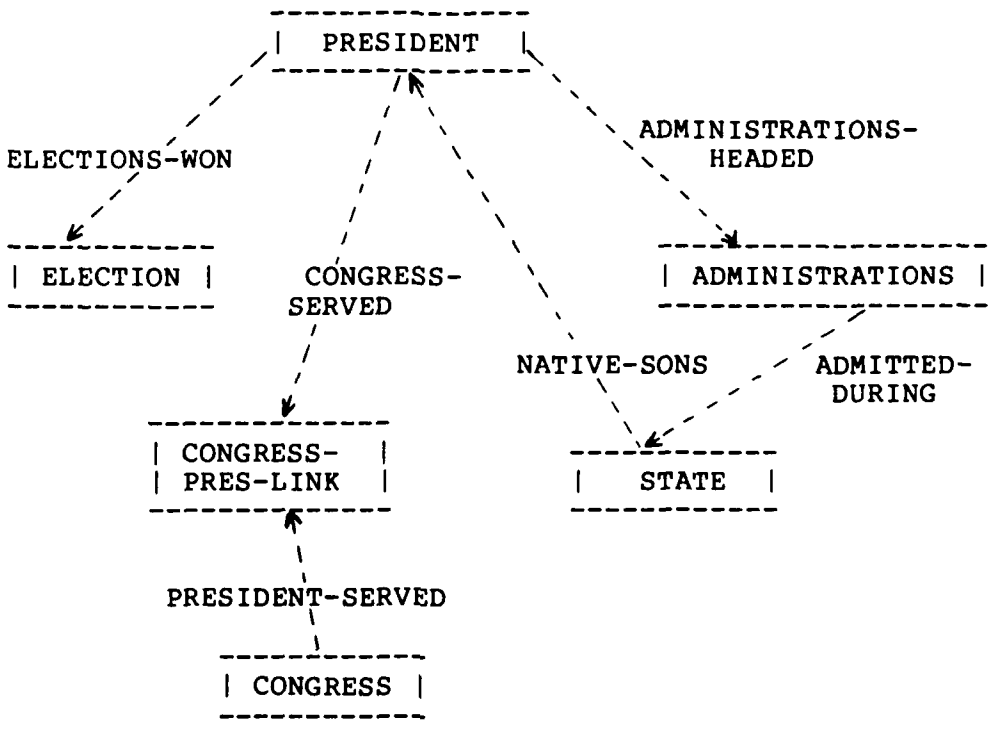


Figure 3. Presidential database data structure diagram without Singular sets.

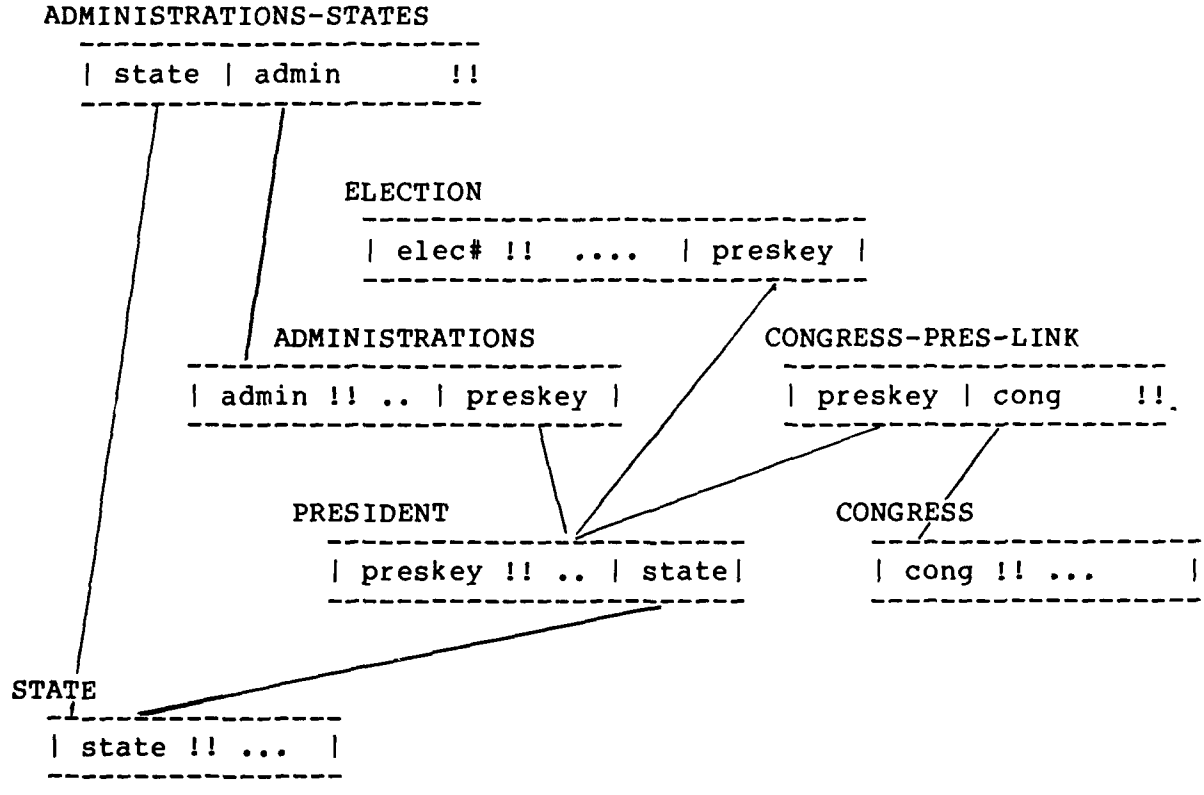


Figure 4. Relational schema of the presidential database (after mapping).

A.4 Schemas

In this section we present the network schema for the presidential database and its corresponding relational schema obtained using the mapping algorithm.

Network Schema (as modelled on DBMS-20):

```
IMAGES BY COMMAND.
NOTE ALL.
INTERCEPT UNANTICIPATED.
JOURNAL JRN1.
RECORDS-PER-PAGE 40.

ASSIGN PRESDB TO PRAREA
BUFFER 4
CALC 40 RPP
FIRST PAGE IS 1
LAST PAGE IS 399
PAGE SIZE IS 512 WORDS.

SCHEMA NAME IS PRDBMS.

AREA NAME IS PRESDB.

RECORD NAME IS PRESIDENT
    LOCATION MODE IS CALC USING PRES-NAME-KEY
    DUPLICATES ARE NOT ALLOWED
    WITHIN PRESDB.

    02 PRES-NAME-KEY PIC A(10) USAGE DISPLAY.
    02 LAST-NAME PIC A(10).
    02 FIRST-NAME PIC A(10).
    02 MIDDLE-INITIAL PIC XX.
    02 PRES-DOB SIZE 15 USAGE DISPLAY.
    02 PRES-HEIGHT PIC X(10) USAGE DISPLAY.
    02 PRES-PARTY PIC A(10) USAGE DISPLAY.
    02 PRES-COLLEGE PIC A(10) USAGE DISPLAY.
    02 PRES-ANCESTRY PIC A(10) USAGE DISPLAY.
```

```
02 PRES-RELIGION PIC A(10) USAGE DISPLAY.
02 PRES-DOD SIZE 15 USAGE DISPLAY.
02 PRES-CAUSE-DEATH PIC X(10) USAGE DISPLAY.
02 PRES-FATHER PIC A(10) USAGE DISPLAY.
02 PRES-MOTHER PIC A(10) USAGE DISPLAY.
02 PRES-NUM-OCCUPATIONS TYPE FIXED BIN REAL.
02 PRES-NUM-MARRIAGES TYPE FIXED BIN REAL.
02 PRES-OCCUPATION SIZE 10 USAGE DISPLAY
    OCCURS 5 TIMES.
02 PRES-MARRIAGE SIZE 27 USAGE DISPLAY
    OCCURS 5 TIMES.

  RECORD NAME IS ADMINISTRATION
      LOCATION MODE IS VIA ADMINISTRATIONS-HEADED
      WITHIN PRESDB.

      02 ADMIN-KEY PIC XXX USAGE DISPLAY.
      02  ADMIN-INAUGURATION-DATE SIZE 15 USAGE DISPLAY.
      02  ADMIN-NUM-VPS TYPE FIXED BIN REAL.
      02  ADMIN-VP SIZE 20 USAGE DISPLAY
              OCCURS 5 TIMES.

  RECORD NAME IS STATE
      LOCATION MODE IS CALC
          USING STATE-NAME DUPLICATES ARE NOT ALLOWED
      WITHIN PRESDB.

      02  STATE-NAME PIC X(10) USAGE DISPLAY.
      02  STATE-YEAR-ADMITTED PIC 9999 USAGE DISPLAY.
      02  STATE-CAPITAL PIC X(10) USAGE DISPLAY.
      02  STATE-AREA-INFO SIZE 8 USAGE DISPLAY.
      02  STATE-POPULATION-INFO SIZE 10 USAGE DISPLAY.
      02  STATE-ELECTORAL-VOTES PIC 999 USAGE DISPLAY.
      02  STATE-NUM-MAJOR-CITIES TYPE FIXED BIN REAL.
      02  STATE-MAJOR-CITY SIZE 18 USAGE DISPLAY
              OCCURS 15 TIMES.

  RECORD NAME IS ELECTION
      LOCATION MODE IS VIA ALL-ELECTIONS-SS
      WITHIN PRESDB.

      02  ELECTION-YEAR PIC 9999 USAGE DISPLAY.
      02  ELECTION-WIN-ELECTORAL-VOTES PIC 999 USAGE DISPLAY.
      02  ELECTION-NUM-LOSERS TYPE FIXED BIN REAL.
      02  ELECTION-LOSER SIZE 23 USAGE DISPLAY
              OCCURS 9 TIMES.

  RECORD NAME IS CONGRESS
      LOCATION MODE IS CALC
          USING CONGRESS-KEY DUPLICATES ARE NOT ALLOWED
```

```
        WITHIN PRESDB.

            02  CONGRESS-KEY PIC X(4) USAGE DISPLAY.
            02  CONGRESS-NUM-PARTY-SENATE TYPE FIXED BIN REAL.
            02  CONGRESS-NUM-PARTY-HOUSE TYPE FIXED BIN REAL.
            02  CONGRESS-PARTY-SENATE SIZE 13 USAGE DISPLAY
                    OCCURS 5 TIMES.
            02  CONGRESS-PARTY-¬HOUSE SIZE 13 USAGE DISPLAY
                    OCCURS 5 TIMES.

    RECORD NAME IS CONGRESS-PRES-LINK
        LOCATION MODE IS VIA CONGRESS-SERVED
        WITHIN PRESDB.

            02  CONGRESS-PRES-LINK-KEY PIC XXXX USAGE DISPLAY.


    SET NAME IS ALL-PRESIDENTS-SS
        MODE IS CHAIN
        ORDER IS SORTED DUPLICATES ARE LAST
        OWNER IS SYSTEM
        MEMBER IS PRESIDENT MANDATORY AUTOMATIC
            ASCENDING KEY IS LAST-NAME, FIRST-NAME.

    SET NAME IS ALL-STATES-SS
        MODE IS CHAIN
        ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
        OWNER IS SYSTEM
        MEMBER IS STATE MANDATORY AUTOMATIC
            ASCENDING KEY IS STATE-NAME.

    SET NAME IS ALL-CONG-PRES-LINKS-SS
        MODE IS CHAIN   LINKED PRIOR
        ORDER IS SORTED DUPLICATES ARE LAST
        OWNER IS SYSTEM
        MEMBER IS CONGRESS-PRES-LINK MANDATORY AUTOMATIC
            ASCENDING KEY IS CONGRESS-PRES-LINK-KEY.

    SET NAME IS ALL-ELECTIONS-SS
        MODE IS CHAIN
        ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
        OWNER IS SYSTEM
        MEMBER IS ELECTION MANDATORY AUTOMATIC
            ASCENDING KEY IS ELECTION-YEAR.

    SET NAME IS NATIVE-SONS
        MODE IS CHAIN
        ORDER IS SORTED DUPLICATES ARE LAST
        OWNER IS STATE
        MEMBER IS PRESIDENT MANDATORY AUTOMATIC LINKED TO OWNER
```

```
            ASCENDING KEY IS LAST-NAME, FIRST-NAME
            SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

        SET NAME IS ADMINISTRATIONS-HEADED
            MODE IS CHAIN
            ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
            OWNER IS PRESIDENT
            MEMBER IS ADMINISTRATION MANDATORY AUTOMATIC LINKED TO OWNER
                ASCENDING KEY IS ADMIN-KEY
                SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

        SET NAME IS ADMITTED-DURING
            MODE IS CHAIN
            ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
            OWNER IS ADMINISTRATION
            MEMBER IS STATE OPTIONAL MANUAL LINKED TO OWNER
                ASCENDING KEY IS STATE-NAME
                SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

        SET NAME IS ELECTIONS-WON
            MODE IS CHAIN   LINKED PRIOR
            ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
            OWNER IS PRESIDENT
            MEMBER IS ELECTION MANDATORY AUTOMATIC LINKED TO OWNER
                ASCENDING KEY IS ELECTION-YEAR
                SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

        SET NAME IS CONGRESS-SERVED
            MODE IS CHAIN
            ORDER IS SORTED DUPLICATES ARE NOT ALLOWED
            OWNER IS PRESIDENT
            MEMBER IS CONGRESS-PRES-LINK MANDATORY AUTOMATIC LINKED TO
            OWNER ASCENDING KEY IS  CONGRESS-PRES-LINK-KEY
                SET OCCURRENCE SELECTION IS THRU LOCATION MODE OF OWNER.

        SET NAME IS PRESIDENT-SERVED
            MODE IS CHAIN
            ORDER IS LAST
            OWNER IS CONGRESS
            MEMBER IS CONGRESS-PRES-LINK OPTIONAL MANUAL LINKED TO OWNER
                SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.


        SUB-SCHEMA NAME IS TOTAL.

        AREA SECTION.
            COPY ALL AREAS.
        RECORD SECTION.
```

```
  01 PRESIDENT.
02 PRES-NAME-KEY.
02 LAST-NAME.
02 FIRST-NAME.
02 MIDDLE-INITIAL.
02 PRES-DOB.
   03 MONTH-B PIC A(9).
   03 DAY-B PIC 99.
   03 YEAR-B PIC 9999.
02 PRES-HEIGHT.
02 PRES-PARTY.
02 PRES-COLLEGE.
02 PRES-ANCESTRY.
02 PRES-RELIGION.
02 PRES-DOD.
   03 MONTH-D PIC A(9).
   03 DAY-D PIC 99.
   03 YEAR-D PIC 9999.
02 PRES-CAUSE-DEATH.
02 PRES-FATHER.
02 PRES-MOTHER.
02 PRES-NUM-OCCUPATIONS.
02 PRES-NUM-MARRIAGES.
02 PRES-OCCUPATION.
   03 OCCUPATION-NAME PIC X(10).
02 PRES-MARRIAGE.
   03 WIFE-NAME PIC A(10).
   03 MARRIAGE-DATE.
      04 MONTH-M PIC A(9).
      04 DAY-M PIC 99.
      04 YEAR-M PIC 9999.
       03 NUM-CHILDREN PIC 99.
  01 ADMINISTRATION.
      02 ADMIN-KEY.
      02  ADMIN-INAUGURATION-DATE.
          03  MONTH PIC A(9).
          03  DAY PIC 99.
          03 YEAR PIC 9999.
      02  ADMIN-NUM-VPS.
      02  ADMIN-VP.
          03  VP-LAST-NAME PIC A(10).
          03  VP-FIRST-NAME PIC A(10).
  01 STATE.
      02  STATE-NAME.
      02  STATE-YEAR-ADMITTED.
      02  STATE-CAPITAL.
      02  STATE-AREA-INFO.
          03  AREA-SIZE PIC 9(6).
          03  AREA-RANK PIC 99.
      02  STATE-POPULATION-INFO.
```

```
            03  POPULATION-SIZE PIC 9(8).
            03  POPULATION-RANK PIC 99.
        02  STATE-ELECTORAL-VOTES.
        02  STATE-NUM-MAJOR-CITIES.
        02  STATE-MAJOR-CITY.
            03  CITY-NAME PIC X(10).
            03  CITY-POPULATION PIC 9(8).
01 ELECTION.
        02  ELECTION-YEAR.
        02  ELECTION-WIN-ELECTORAL-VOTES.
        02  ELECTION-NUM-LOSERS.
        02  ELECTION-LOSER.
            03  LOSER-LAST-NAME PIC X(10).
            03  LOSER-PARTY PIC A(10).
            03  LOSER-ELECTORAL-VOTES PIC 999.
01 CONGRESS.
        02  CONGRESS-KEY.
        02  CONGRESS-NUM-PARTY-SENATE.
        02  CONGRESS-NUM-PARTY-HOUSE.
        02  CONGRESS-PARTY-SENATE.
            03  PARTY-NAME-S PIC A(10).
            03  NUM-SEATS-S PIC 999.
        02  CONGRESS-PARTY-HOUSE.
            03  PARTY-NAME-H PIC A(10).
            03  NUM-SEATS-H PIC 999.
01 CONGRESS-PRES-LINK.
        02  CONGRESS-PRES-LINK-KEY.
SET SECTION.
        COPY ALL SETS.

END-SCHEMA.
```

Relational schema (as modelled on Datacomputer):

```
CREATE PRESIDENT FILE LIST(,100)
    PRESRECORD STRUCTURE
    PRESNAMEKEY STR(10)    /* Key component */
    NAME STRUCTURE
        LASTNAME STR(10)
        FIRSTNAME STR(10)
        MIDDLEINITIAL STR(1)
    END
    PRESDOB STRUCTURE
        MONTHB STR(9)
        DAYB INT (2)
        YEARB INT (4)
    END
    PRESHEIGHT STR(10)
    PRESPARTY STR(10)
    PRESCOLLEGE STR(10)
    PRESANCESTRY STR(10)
    PRESRELIGION STR(10)
    PRESDOD STRUCTURE
        MONTHD STR(9)
        DAYD INT (2)
        YEARD INT (4)
    END
    PRESCAUSEDEATH STR(10)
    PRESFATHER STR(10)
    PRESMOTHER STR(10)
    PRESNUMOCCUPATIONS INT(1)
    PRESNUMMARRIAGES INT(1)
    PRESOCCUPATION LIST (,5),C=PRESNUMOCCUPATIONS
        OCCUPATIONNAME STR(10)
    PRESMARRIAGE LIST (,5),C=PRESNUMMARRIAGES
        MARRIAGEINFO STRUCTURE
        WIFENAME STR(10)
        MARRIAGEDATE STRUCTURE
            MONTHM STR(9)
            DAYM INT(2)
            YEARM INT(4)
        END
        NUMCHILDREN INT(2)
        END
    STATENAME STR(10)     /* Foreign Key */
    END;

CREATE ADMINISTRATION FILE LIST(,100)
    ADMINRECORD STRUCTURE
        ADMINKEY STR(2)    /* Key component */
    PRESNAMEKEY STR(10)  /* Foreign key */
```

```
        ADMININAUGURATIONDATE STRUCTURE
            MONTH STR(9)
            DAY INT(2)
            YEAR INT(4)
    END
        ADMINNUMVPS INT(1)
        ADMINVP LIST(,5),C=ADMINNUMVPS
        VPNAME STRUCTURE
                VPLASTNAME STR(10)
                VPFIRSTNAME STR(10)
        END
    END;

CREATE STATE FILE LIST(50)
    STATERECORD STRUCTURE
        STATENAME STR(10)    /* Key component */
        STATEYEARADMITTED INT(4)
        STATECAPITAL STR(10)
        STATEAREAINFO STRUCTURE
            AREASIZE INT(6)
            AREARANK INT(2)
    END
        STATEPOPULATIONINFO STRUCTURE
            POPULATIONSIZE INT(8)
            POPULATIONRANK INT(2)
    END
        STATEELECTORALVOTES INT(3)
        STATENUMMAJORCITIES INT(2)
        STATEMAJORCITY LIST(,15),C=STATENUMMAJORCITIES
        CITYRECORD STRUCTURE
            CITYNAME STR(10)
            CITYPOPULATION INT(8)
        END
    END;

CREATE ELECTION FILE LIST (,100)
    ELECTRECORD STRUCTURE
        ELECTIONYEAR INT(4)  /* Key component */
    PRESNAMEKEY STR(10)  /* Foreign key */
        ELECTIONWINELECTORALVOTES INT(3)
        ELECTIONNUMLOSERS INT(2)
        ELECTIONLOSER LIST(,9),C=ELECTIONNUMLOSERS
        LOSERINFO STRUCTURE
                LOSERLASTNAME STR(10)
                LOSERPARTY STR(10)
                LOSERELECTORALVOTES INT(3)
        END
    END;
```

```
CREATE CONGRESS FILE LIST(,150)
    CONGRECORD STRUCTURE
        CONGRESSKEY STR(4)  /* Key component */
        CONGRESSNUMPARTYSENATE INT(1)
        CONGRESSNUMPARTYHOUSE INT(1)
        CONGRESSPARTYSENATE LIST(,5),C=CONGRESSNUM
        SENATEINFO STRUCTURE
                PARTYNAMES STR(10)
                NUMSEATSS INT(3)
        END
        CONGRESSPARTYHOUSE LIST(,5),C=CONGRESSNUMP
        HOUSEINFO STRUCTURE
                PARTYNAMEH STR(10)
                NUMSEATSH INT(3)
        END
    END;

CREATE CONGRESSPRESLINK FILE LIST(,100)
        CONGRESSPRESLINKKEY STRUCTURE  /* Key compo
        PRESNAMEKEY STR(10)  /* Foreign Key */
        CONGRESSKEY STR(4)   /* Foreign key */
    END;

CREATE ADMINISTRATIONSTATES FILE LIST (,50)
    ADMINSTATEKEY STRUCTURE  /* Key component */
        ADMINKEY STR(2)    /* Foreign Key */
        STATENAME STR(10)  /* Foreign Key */
    END;
```

B. DOCFile Queue Description

This appendix describes the queue-level interface to the DOCFile System, a Datacomputer application by which ARPA personnel and contractors can manage a database of official ARPA documents. An operator at a terminal on a TENEX or TOPS-20 system invokes the USER program, which verifies and accepts the operator's requests and appends request records to the DOCFile queue for batch processing by the separate DEMON program. This document supersedes all previous queue documentation.

The DOCFile Queue is a disk file to which everyone has read and append access and to which the DEMON has full access. DOCFile subscribers, using the USER or a comparable program, append their requests directly to the queue.

The queue is simply a concatenation of records: one overall queue header followed by request records. To minimize problems resulting from system crashes, etc., requests are split up into a series of subrequests,

none of which involves more than a single Datacomputer interaction that modifies Datacomputer files. Each request is a concatenation of one request header followed by the various subrequests. Each subrequest is a concatenation of a subrequest header followed by a trailer whose content and format depend on the type of subrequest.

The formats of the various queue components are here represented by BCPL structure declarations. They are the formats current at the time of this writing and are included in this document for the reader's convenience.

The allowable values for various fields are discussed, but note that the data types and field dimensions are ultimately determined by the definitions of the files and ports in the Datacomputer.

Note that the TENEX/TOPS-20 internal date/time format is used in all fields whose value is a timestamp.

B.1 The Queue Header

The QUEUE HEADER:

```
structure {qh queueHEADER     //first words of a queue file

        { format word         //format version number
          numrq word          //number of requests
          dtcreated word      //when file created
          filenum word        //queue file version number
          host word           //host number
          dtreplaced word     //when next file created
          z^10 word           //reserved
        } }qh

manifest { queueHEADERsize := csize queueHEADER }
```

In terms of the current implementation, "dtcreated" and "host" are the only fields with meaningful values.

B.2 The Request Header

The REQUEST HEADER

```
structure {rqh requestHEADER    // first words of a request

{ length        word    // minus total length of request
  hdlength      word    // minus length of request header
                        // (words to start of 1st subrequest)
  type          word    // type of request
  numsbrq       word    // minus number of subrequests in rq
  priority       bit 7  // L, M, or H
  status         bit 28 // 4 ASCII chars giving rq status
        fill     word
  dtappended    word    // when request appended to queue
  dtfirst       word    // when first subrq started
  dtlast        word    // when last subrq ended
  ttynum        word    // submittor's tty number
                        //BCPL strings:
  ldsiz         byte    //login directory name: char count
  ldch ^ 511    char    //login directory name: 9-bit chars
//cnsiz byte            //connected dir name: char count
//cnch ^ 511            //    "      "   "    9-bit chars
} }rqh
```

```
//Maximum request header size:

manifest { requestHEADERsize := csize requestHEADER + 128 }
```

Field	Value
"priority"	One ASCII capital letter: 'L', 'M', or 'H' for "low", "medium", or "high".
"status"	Four ASCII characters indicate disposition of request as follows: 'NULL' initial value from user program 'BFMT' bad format, request ignored 'LOSE' all of request failed 'WINS' all of request successful 'CAND' request cancelled 'MIXD' subrequest dispositions mixed

B.3 The Subrequest Header

The SUBREQUEST HEADER

```
structure {sbrqh subrqHEADER    //first words of subrequest

{ length        word    //negative word count in subrequest
  status         bit 28 //really 4 ascii char status
  retry          bit 7  //retry count
    fill         word
  qid           word    //see Qid structure definition
  dtaccept      word    //when subrequest first tried
  dtdone        word    //when subrequest completed
  value         word    //value returned by subrq, if any
  spare         word    //spare
  type          word    //really 5 ascii chars for subrq type
} }sbrqh

//number of words in subrequest header:

manifest { subrqHEADERsize := csize subrqHEADER }
```

Unless otherwise stated, subrequest header fields may be initialized to zero.

field value

"retry" To be written by DEMON; should be initialized to zero by the USER program.

"status" Subrequest disposition; same codes as are used in the request header.

"value" To be written by DEMON. This value is derived from the execution of the subrequest, and may be the index for a new contractor, the DOCFile document number for a stored text, etc.

"type" Five ASCII letters for each subrequest type:

'CANCL' Cancel a queued request.

'NCONT' Add new contractor name.

'NPROG' Add new program name.

'RETRE' Retrieve headers/documents.

'STORH' Store new document header.

'STORT' Store new document text.

'UPDAT' Update document header(s).

B.4 The Subrequest Trailers

The SUBREQUEST TRAILERS

Arguments for each type of subrequest follow the Subrequest Header in a variable-length field formatted as follows:

CANCL: One word: the number of the request to be cancelled. The request immediately following the queue header is request number one.

NCONT: Contractor name as a BCPL string.

NPROG: Program name as a BCPL string.

RETRE: The USER program prompts the operator for a specification of the set of documents or headers to be retrieved. This spec has the form of a Boolean constraint on the values of various document attributes, i.e. fields within the document header. USER then passes to DEMON both a human-oriented string and a syntactically correct Datalanguage "WITH" clause representing this specification. The

"WITH" clause includes the initial word "WITH" but does not include the semicolon that would terminate the Datalanguage request in which this "WITH" clause will be embedded.

In the current implementation, both strings are in BCPL string format, but the 511-character size limit on such strings is really too small.

The "subtype" field contains one uppercase ASCII letter: 'H' implies that only headers are to be retrieved, 'D' causes both headers and their associated documents to be retrieved.

```
structure
 {retup retupTRAILER
{ NonDLDisplacement     word    // ... to plain string
  DLDisplacement        word    // ... to DL "WITH" clause
  lastDisplacement      word    // ... to "last" field
  subtype               char
    fill word
  //NonDLSize           byte    // human-oriented BCPL string
  //NonDLCh^511         char
  //DLSize              byte    // Datalanguage BCPL string
  //DLCh^511            char
  //lastsize            byte    // BCPL string: dest file name
  //lastch^511          char
}
 }retup

manifest { retupTRAILERsize := csize retupTRAILER + 384 }
```

STORH: Arguments corresponding to the fields in the HEADERS file in the Datacomputer.

```
structure {addhd storhTRAILER

{ pub                        bit    //public bit
  del                        bit    //deleted bit
  sparel                     bit 16 //spare field
  dtype                      bit 7  //document type
    fill                     word
  docid                     word    //document id
  contr                     word    //contractor number
  arpaprog                  word    //program number
  arpao                     word    //arpa order number
  arpal                     word    //arpa line number
  contract^maxcontract byte         //contract number
    fill                     word
  ddate                     word    //document date
  progsys^maxprogsys         char   //programming system
  oprsys^maxoprsys           char   //operating system
  progsysv^maxprogsysver char       //programming system version
  oprsysv^maxoprsysver       char   //operating system version
    fill                     word
  titleDisplacement         word    //   word displacements
  authorsDisplacement       word    //   within the trailer
  abstractDisplacement      word    //   of various
```

```
  wordsDisplacement     word    //   strings
  //titlesize byte             //title length (BCPL string)
  //titlech^maxDCtitle char    //title characters
                               // BigCPLString strings:
  //authorssize word           //authors length
  //authorsch^maxDCauth char   //authors characters
  //abstractsize word          //abstract length
  //abstractch^maxabstract char //abstract characters
  //words word                 //keywords length
  //wordsch^maxDCwords char    //keywords characters
} }addhd

manifest { storhTRAILERsize := csize storhTRAILER
                             + 3
                             + ( maxDCtitle
                                 + maxDCauth
                                 + maxDCwords
                                 + maxabstract
                                 + 12
                               ) / 4
         }
```

"public" and "deleted" are Boolean flag fields in which 0 means "false".

"dtype" is an uppercase ASCII letter, which currently may be one of:

"F"	Final report
"M"	M.R.A.O.
"P"	Proposal
"S"	Software source
"T"	Technical report

"docid", "contr", and "arpaprog" may be zero, in which case their values must be derived by the DEMON from the results of the previously-executed STORT, NCONT, or NPROG subrequest, respectively, in this same request.

AUTHORS, KEYWORDS, and ABSTRACT are 9-bit character strings whose counts are contained in full words according to the structure "BigCPLString".

The AUTHORS list is a concatenation of individual author names, each of the form "<LASTNAME><space><FULLNAME><;>". The full name may contain space, comma, and period characters, but the last name may not contain a space.

KEYWORDS may contain spaces, and each keyword (or key phrase) in the concatenated list must be terminated by a Control-A character (octal 001).

STORT: A ASCIZ string which is the filename of the text.

UPDAT: This subrequest is not implemented.

C. Pascal Interface to Datacomputer

Simulation of Fortran calling conventions:

In general, in Pascal, an external procedure is declared as follows:

```
Procedure declaration; EXTRN;
```

For example,

```
procedure P (var X,Y : integer); EXTRN;
```

P should then use Pascal calling conventions. However, should we prefer Fortran calling conventions, the routine P should be declared EXTRN FORTRAN. That is,

```
procedure P (var X,Y : integer); EXTRN FORTRAN;
```

This would cause the Pascal compiler to expect a routine P that uses Fortran calling conventions.

Use of DCPKG from Pascal programs:

By declaring all the entry points of DCPKG as EXTRN FORTRAN in the Pascal application program, the Pascal program can access the Datacomputer as if it were a Fortran program. The following is a Pascal application program illustrating the use of DCPKG to access the Datacomputer.

```
(*D-*)    Program PASDC (INPUT);
(* A program to access the Datacomputer. This program
   uses the DCPKG routines by simulating the Fortran
   calling convention from the Pascal program
 *)

LABEL   99;

CONST

        stringlength = 256;

TYPE

        string  = Packed Array [1..stringlength] of CHAR;
        data    = Packed Array [1..80] of CHAR;

VAR

        host,socket,retcode: integer;
        name    : packed array [1..4] of CHAR;
        command : string; length : integer;
        line    : data;
        listfile: file of CHAR;
        cnum    : integer;

(* The following are declarations of procedures which
   are external to this program. EXTERN FORTRAN simulates
   Fortran calling convention.
```

```
     The following procedures are entry points into the
     DCPKG package.
   *)

Procedure DCSTRT (VAR host,socket,retcode : integer);
                                          EXTERN FORTRAN;
(* Establishes the Datacomputer connections *)

Procedure DCCMDS (VAR command: string; VAR cmdlen : integer);
                                          EXTERN FORTRAN;
(* Takes a Datalanguage command and ships it to the Datacomputer *)

Procedure DCREAD (VAR line: data; VAR cnum: integer);
                                          EXTERN FORTRAN;
(* Reads the data sent by the Datacomputer *)

Procedure DCSTOP; EXTERN FORTRAN;
(* Terminates Datacomputer connections *)

(*******************************************************)
Procedure GETCOMMAND;
(* reads a Datalanguage command from the input file *)
(*******************************************************)
var i:  integer;
begin
        read (INPUT, command : length);
        readln(INPUT);
        for i := 1 to length do write (TTY, command[i]);
```

```
        writeln (TTY);
end;  (* getcommand *)
(*************************************************************)
Procedure CLEARARRAY;
(* initialize array line with blanks *)
(*************************************************************)
var i:  integer;
begin
        for i := 1 to 80 do line[i] := ' ';
end;  (* cleararray *)
(*************************************************************)
(* sample interface program begins here *)
BEGIN   (* main program *)
        write (TTY, 'please type user name:');
        readln (TTY); read (TTY,name);
        if name <> 'nori' then begin
                        writeln (TTY, 'not a valid user');
                        goto 99;
                        end;
      cnum    := 0;
      host    := 31;  (* CCA *)
      socket  := 131;
      DCSTRT (host,socket,retcode);
      if retcode = 1 then begin
```

```
                writeln (TTY, 'Datacomputer cannot be started');
                DCSTOP; goto 99;
                end;
        getcommand; (* login command *)
        (* LOGIN CCA.NORI; *)
        DCCMDS (command, length);
        getcommand;
        (* OPEN PRESIDENT; LIST PRESIDENT %SOURCE; *)
        DCCMDS (command, length); (* list command *)
        if retcode <> 2 then begin
                        writeln (TTY, retcode:3);
                        DCSTOP; goto 99;
                        end;
        (* transfer start *)
        rewrite (listfile, 'pres.lst;P775252');
        loop
                cleararray; (* clear line array *)
                DCREAD (line,cnum);
                writeln (listfile, line);
                EXIT if retcode = 2;
        end;
        getcommand;
        (* CLOSE %OPEN; *)
        DCCMDS (command, length);
        writeln (TTY, 'successful');
```

```
        DCSTOP;
99:     (* programs ends *)
END.
```

D. MBQ Program

MBQ is a program designed to ease the process of sending a group of messages to MARS-Filer from Tenex and TOPS20 systems on the ARPAnet. The use of MBQ ensures that these "batches" will arrive in the proper format so that they can be processed correctly.

To run the program, the user types "MBQ". The version number and a short description of the program is then displayed on the screen, and followed by a prompt for the name of the message file to be queued. A more verbose description of the program can be displayed in response to a "?" typed by the user. The user is then prompted once again for a file name. After typing in the file name, the user is then asked if the messages are to be filed as "private" mail.

"Private" mail is defined as mail which may be retrieved by those persons named on the distribution list of the original message and by the person who composed the message and s/he who archived it. Most messages are filed as private mail.

Public mail, on the other hand, is mail which can be retrieved by any member of the ARPAnet user community who has access to mail-handling facilities.

The program works by creating a new copy of the message file, naming it "[--UNSENT-MAIL--].MARS-Filer@CCA". The messages are bound in an "envelope" which contains message-header information for use by MARS-Filer in indexing the messages. The file is left queued for delivery to the MARS-Filer mailbox by the local system's MAILER program.

E. MARS Retrieval Specifications

Message retrievals are initiated by sending a Retrieval Request (RR), which is a specially formatted ARPAnet message, to MARS-Retriever@CCA.

Retrieved messages will be mailed back, one at a time, and will appear as new mail in the requester's mailbox.

The following example will retrieve any RECENT PUBLIC message with MSGGROUP and 1001 in the SUBJECT field, including MSGGROUP#1001.

```
***** begin sample RR
Date: 28 DEC 1979 20:20-PST
From: ESTEFFERUD at USC-ECL
Subject: RECENT PUBLIC
To:   MARS-R@CCA
cc:   JZS@CCA

SUBJECT: MSGGROUP 1001

-------
***** end sample RR
```

The following is a sample RR message which will retrieve all mail sent FROM: Panko TO: MsgGroup during the month of January 1978.

```
***** begin sample RR
From: JZS at CCA (Joanne Z. Sattley)
Subject: All Public
To: MARS-Retriever at CCA
cc: JZS at CCA

AFTER: 31-DEC-77
BEFORE: 1-FEB-78
TO: MSGGROUP
FROM: PANKO

-------
***** end sample RR
```

RR (Retrieval Request) messages can be composed according to the following specification using any ARPAnet message composition program:

- The TO field of the RR message must contain

  "MARS-Retriever@CCA"

- The SUBJECT field of the RR message should contain one choice from each of the following keyword pairs:

ALL or RECENT; PUBLIC or PRIVATE.

RECENT means since 1 January 1979, & ALL includes RECENT

Omission of ALL defaults to RECENT.

Omission of PUBLIC defaults to PRIVATE.

Omission of both defaults to RECENT PRIVATE.

. The BODY of the RR message will be interpreted by MARS-Retriever to perform the retrieval. Its content and format should correspond to an ARPAnet message header, or selected portions thereof.

. The following list specifies which field names will be recognized in the RR message body by MARS-Retriever@CCA to establish search criteria, with some notes on their interpretation.

. The scanning of each field is terminated by a carriage-return.

DATE: This field ill cause only those messages DATED on the specified date to be retrieved. The format of the date field is day-month-year. Use of hyphens is optional.

AFTER: Use of this field will retrieve messages DATED after the specified date, and including the specified date.

SINCE: This field is interpreted exactly like the AFTER: field.

BEFORE: Use of this field will retrieve messages DATED before the specified date, but not including the specified date.

UNTIL: This field is interpreted exactly like the BEFORE: field.

FROM: This field is expected to contain a valid mailbox name, but validity is not checked. Host specification is optional, but if included it must be separated from the name by the string " at " or by an @ sign. If more than one name is specified, ORing of the names is implicit for this field. Retrieval based upon host specification alone has not been implemented.

TO: This field is expected to contain one or more valid mailbox names, but validity is not checked. Host specification is optional, but if included it must be separated from the name by the string " at " or by an @ sign with no adjacent spaces. Spaces and commas between mailbox names imply AND for this field.

Retrieval based upon host specification alone has not been implemented.

SUBJECT: Use of this field will retrieve all messages whose SUBJECT field contents match the specified word(s). Spaces and commas imply AND. The use of OR must be explicit.

TEXT: Use of this field will retrieve all messages whose TEXT body content contains matches for the specified word(s). Spaces and commas imply AND. Use of OR must be explicit.

EXAMPLE RETRIEVAL FIELDS:

```
SUBJECT: MSGGROUP 1001 or MSGGROUP#1001  ; OR must be explicit

TEXT: FINGER privacy,community           ; spaces & commas imply AND

DATE: 14 March 1978

SINCE: 1 Dec 78                          ; same as AFTER: 1 Dec 78
AFTER: 1  Dec 1978

UNTIL: 15 March  1979                    ; same as BEFORE: 15 March 1979
BEFORE: 15-MAR-79
```

```
FROM:  JZS at CCA                    ; host specificatio
FROM:  Stef,JZS                      ; comma implies OR

TO: MSGGROUP@ECL                     ; host specificatio
TO: MsgGroup at ECL                  ; "@" or " at " may

TO: MsgGroup,Header-People           ; spaces and commas
```

References

[BACHMAN]
Bachman, C.W. "Data Structure diagrams," Database 1, 2 (summer 1969).

[BBN]
BBN Report No. 3460, "VELANET Communication Protocols between the CCP and the SIP and between the CCP and the DP", 5 April 1977.

[BORKIN]
Borkin, S. "Data model equivalence," proc. Fourth Intl. Conf. on Very Large Data Bases, Berlin, Sept. 1978, pp. 526-534.

[CCA]
Computer Corporation of America, Datacomputer Version 5 User Manual, Cambridge, Massachusetts, July 1978.

[DATE]
Date, C.J. "An Introduction to Database Systems," Addison-Wesley Publishing Company.

[DEC]
"Data Base Management System Administrator's Procedures Manual," order # AA-4146B-TM, Digital Equipment Corporation, Massachusetts.

[DORIN & EASTLAKE]
R. H. Dorin and Donald E. Eastlake, III; "Use of the Datacomputer in the Vela Seismological Network"; Technical Report No. CCA-77-08, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts, 02139, April 15, 1977.

[DORIN & SATTLEY]
Dorin, R.H. and Sattley, J.Z. Very Large Databases: Final Technical Report, Technical Report No. CCA-77-10, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts, 02139, August 30, 1977.

[EASTLAKE a]
Eastlake, III, D.E. "Tertiary Memory Access and Performance in the Datacomputer", Technical Report No. CCA-77-11, Computer Corporation of America,

575 Technology Square, Cambridge, Massachusetts, 02139, June 30, 1977.

[EASTLAKE b]
Eastlake, III, D.E., "SIP Accommodation of Additional Seismic Data Sites", 29 January 1979, Technical Report CCA-79-08, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[EASTLAKE & SATTLEY a]
Eastlake, III, D.E., and Sattley, J.Z., "Program Design for SWF-D: The Signal Waveform File Demon", 30 June 1978, Technical Report CCA-78-10A, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[EASTLAKE & SATTLEY b]
Eastlake, D.E., III and Sattley, J.Z., "Assistance in the Design of the SWF System", January 30, 1979, CCA-79-13, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[KAY]
Kay, M.H., "An Assessment of the CODASYL DDL for Use with a Relational Subschema," in Data Base Description, (B.C.M. Douque and G.M. Nijssen, eds.), North-Holland Publishing Company, Amsterdam, 1975.

[MCGEE]
McGee, W.C., "A Contribution to the Study of Data Equivalance," in Data Base Management, (J.W. Klimbie and K.L. Koffeman, eds.), North-Holland Publishing Company, New York, 1974.

[MARILL & STERN]
Marill, Thomas; and Stern, D.H. "The Datacomputer: A Network Data Utility", Proceedings AFIPS National Computer Conference, AFIPS Press, Vol. 44, 1975.

[SATTLEY a]
Joanne Z. Sattley, "Report on the Implementation and Test of SWF-D: The Signal Waveform File Demon", Technical Report CCA-79-09, 31 January 1979, Computer Corporation of America, 575 Technology Square, Cambridge, Massachusetts 02139.

[SATTLEY b]
Joanne Z. Sattley, "MARS - A Message Archiving and Retrieval Service", 8 January 1978, ARPA Network Working Group Request for Comments #744, Network Information Center #42857.

[TAYLOR & FRANK]
Taylor, Robert and Frank, Randall, "CODASYL Data Base Management Systems," in Computing Surveys, Vol. 8, No. 1, March 1976. pp. 67-103.